Vulnerabilidades no *software* da urna eletrônica brasileira

Diego F. Aranha[1], Marcelo Monte Karam[2], André de Miranda[2], Felipe Scarel[2]

[1]Departamento de Ciência da Computação – Universidade de Brasília (CIC/UnB)
[2]Centro de Informática – Universidade de Brasília (CPD/UnB)

Versão 1.0.2

31 de Março de 2013

[1]Coordenador da equipe.

Resumo

Este relatório apresenta uma análise de segurança do *software* da urna eletrônica brasileira baseada na experiência dos autores enquanto participantes da 2ª edição dos Testes Públicos de Segurança do Sistema Eletrônico de Votação organizados pelo Tribunal Superior Eleitoral (TSE). Durante o evento, foram detectadas vulnerabilidades no *software* que permitiram a recuperação em ordem dos votos computados. Apresentamos cenários onde as vulnerabilidades permitem a possibilidade de fraude eleitoral e sugestões para se restaurar a segurança dos mecanismos afetados. Também são apontadas outras fragilidades no *software* e nas práticas utilizadas para confecção do mesmo. Em particular, este relatório versa sobre os principais problemas de projeto e/ou implementação de mecanismos de segurança detectados no *software* da urna eletrônica:

- **Proteção inadequada do sigilo do voto:** os votos são armazenados fora de ordem, mas é trivial recuperá-los em ordem a partir unicamente dos produtos públicos de uma eleição e conhecimento superficial do código-fonte, também de acesso público aos partidos políticos;

- **Cifração inadequada:** a mesma chave criptográfica é utilizada para cifrar as mídias de todas as urnas eletrônicas. Utilizando a analogia clássica de um cadeado como abstração de técnica criptográfica, isto é equivalente a proteger meio milhão de cadeados com uma mesma chave, visto ser este o número aproximado de equipamentos em operação. Além disso, a chave que decifra todas as mídias é armazenada às claras na porção decifrada das mídias. Utilizando a mesma analogia, isto equivale a esconder a chave do cadeado embaixo do tapete e confiar no segredo dessa localização como fonte de segurança;

- **Utilização de algoritmos obsoletos:** a função de resumo criptográfico utilizada não mais oferece a segurança esperada para aplicação em verificação de integridade. Esta aplicação específica da função escolhida não é mais recomendada há pelo menos 6 anos;

- **Formulação equivocada do modelo de atacante:** há ênfase demasiada no projeto de mecanismos resistentes apenas a atacantes externos, quando agentes internos representam risco muito maior;

- **Processo de desenvolvimento defeituoso:** práticas inseguras permitem a inserção acidental ou maliciosa de vulnerabilidades de *software*, claramente atestando que o processo de desenvolvimento adotado pelo TSE é imaturo do ponto de vista de segurança;

- **Verificação insuficiente de integridade**: o *software* da urna eletrônica verifica sua própria integridade durante o processo de inicialização, mas toda a informação necessária para subverter esse mecanismo encontra-se armazenada nas próprias urnas eletrônicas, com dificuldades distintas para um ataque, dependendo da presença do módulo de segurança em *hardware*. Em urnas sem este recurso, o problema de verificação é reduzido a si próprio, sem fonte externa de confiança. Nesse caso, "*software* auto-verificável" [1] por assinatura digital equivale a confiar a autenticidade de uma assinatura de punho em um documento apenas ao testemunho do próprio "autor", que, assim, pode se passar por quem quiser. É importante ressaltar ainda que uma assinatura autêntica apenas atesta o processamento do conteúdo assinado em algum ponto no tempo e espaço no qual também estava presente a chave de assinatura. Mesmo que os mecanismos de verificação de integridade não sejam contornados e funcionem a contento, ainda não há qualquer garantia de que o conteúdo do documento é de fato o desejado, visto que no caso o mesmo (*software*) tem extensão da ordem de milhões de linhas (de código). Caso o *software* possua vulnerabilidades (como as descritas neste documento), a verificação de integridade (quando não subvertida) tem o efeito colateral de garantir que as mesmas vulnerabilidades estarão presentes em todas as urnas. A versão do código observada pelos autores apresentava ainda como desativada a verificação de integridade de parte do *software* contido na urna, evidenciando as limitações intrínsecas da técnica.

Mais detalhes a respeito dos problemas acima são fornecidos no decorrer deste relatório, mas pode-se observar de antemão que vários dos recursos implementados no *software* da urna eletrônica não representam mecanismos de *segurança*, mas apenas de *ofuscação*, não resistindo a colaboradores internos ou atacantes persistentes. Como vários dos problemas encontrados resultam de falhas arquiteturais ou premissas inadequadas de projeto, é improvável que a intervenção pontual em algumas dessas questões resolva as causas fundamentais para a sua ocorrência. É imprescindível que se execute revisão crítica completa dos processos de desenvolvimento de *software* para que se estabeleçam boas práticas que tenham condições de evitar que novas vulnerabilidades sejam inseridas acidentalmente ou intencionalmente por agentes maliciosos internos ou externos. Como o modelo de urna eletrônica adotado no Brasil depende exclusivamente da integridade do *software* para se atingir integridade dos resultados, os problemas discutidos aqui adquirem uma caráter crítico e exigem urgência na introdução de mecanismos que permitam a auditabilidade de resultados independente do *software*. Apenas com uma revisão de práticas e instalação de metodologia científica para avaliação contínua do sistema, é possível que o *software* da urna eletrônica satisfaça requisitos mínimos e plausíveis de segurança e transparência.

Sumário

Capítulo 1

Introdução

O Brasil vem adotando crescente informatização das suas eleições desde o ano de 1996, culminando no cenário atual onde se vislumbra a instalação de dispositivos de identificação biométrica em todos os equipamentos de votação. Marcos importantes na história da iniciativa foram a realização das primeiras eleições puramente eletrônicas em 2000, a transferência da responsabilidade exclusiva do desenvolvimento de *software* para o TSE a partir de 2006 e a adoção de um sistema operacional de código aberto (GNU/Linux) a partir de 2008. Ao se estabilizar os componentes básicos do sistema eletrônico de votação e procedimentos relacionados, entende-se que a preocupação direta deve ser focada no incremento da segurança para que seja possível executar eleições confiáveis que conservem absolutamente o sigilo e a integridade das escolhas definidas pelo eleitor.

Uma iniciativa louvável nesta direção é a realização desde 2009 de testes periódicos e públicos de segurança que permitem, ainda que com algumas restrições indesejáveis, a equipes de especialistas da academia e indústria avaliar de forma independente a segurança dos mecanismos adotados pelo sistema eletrônico de votação.

1.1 Objetivo deste relatório

O objetivo geral do relatório é formalizar as observações realizadas pela equipe de autores quando de sua participação na 2ª edição dos Testes Públicos de Segurança organizados pelo Tribunal Superior Eleitoral (TSE). Os relatórios que foram produzidos em conjunto com o Tribunal, e por ele publicados a título de resultados, carecem de informações sobre outros problemas encontrados que não se relacionam diretamente aos planos de teste formulados e executados pela equipe. Assim, a motivação principal para apresentar este relatório é delinear as limitações do sistema eletrônico de votação adotado no Brasil e contribuir para a evolução do seu processo de segurança. Seguindo políticas padronizadas de divulgação de vulnerabilidades utiliza-

das na área de Segurança da Informação, são apresentadas aqui descrições suficientes das fragilidades e problemas de processo encontrados, acompanhadas de múltiplas sugestões de correção. Desta forma, a parte interessada encontra-se em posição adequada para implementar contra-medidas efetivas.

É importante salientar ainda que o presente relatório trate apenas do *software* da urna eletrônica, não se manifestado a respeito dos aspectos físicos ou do *hardware* do equipamento. Esta decisão foi tomada respeitando-se os campos de especialidade dos autores. Ainda assim, também vale ressaltar que as observações coletadas referem-se apenas a uma pequena – ainda que estratégica – fração do código-fonte do *software*, excluídos também os componentes de *software* que constituem o sistema de votação do qual a urna faz parte, visto que as regras do evento, e o limite de tempo na participação dos investigadores, não permitiram uma avaliação mais detalhada. Por fim, o conteúdo e as conclusões aqui apresentados são de inteira responsabilidade dos autores e não representam de forma alguma a opinião da Universidade de Brasília ou quaisquer outros órgãos aos quais eventualmente os autores prestaram ou venham a prestar serviços.

1.2 Visão superficial do sistema

A urna eletrônica brasileira pode ser classificada como um modelo do tipo *DRE* (Direct Recording Electronic), sem impressão do voto. Em termos gerais, uma eleição utilizando o sistema eletrônico de votação segue as seguintes etapas de preparação:

1. Lacração dos componentes de *software* e produção de mídias de carga;
2. Instalação do *software* nas urnas eletrônicas com mídias de carga;
3. Distribuição das urnas às respectivas seções eleitorais.

No dia determinado para realização das eleições, cada urna eletrônica deve executar uma seqüência bem-definida de procedimentos:

1. Impressão da *zerésima*, documento oficial que supostamente atesta que nenhum voto foi previamente computado para qualquer candidato;
2. Abertura da votação pelo mesário responsável;
3. Acesso dos eleitores à urna eletrônica para que suas escolhas sejam inseridas;
4. Encerramento da votação, realizada também pelo mesário responsável;
5. Emissão de vias do Boletim de Urna (BU) em papel, contendo a totalização parcial dos candidatos;

6. Gravação autenticada dos produtos públicos de votação, abrangendo principalmente as versões digitais do BU, arquivo de registro cronológico de eventos (LOG) e Registro Digital do Voto (RDV);
7. Rompimento do lacre e retirada pelo mesário da Mídia de Resultados (MR) contendo os produtos públicos da eleição;
8. Transmissão dos produtos públicos para o totalizador a partir de rede privada de comunicação.

O papel do totalizador consiste em combinar todas as totalizações parciais no resultado declarado oficial das eleições.

1.3 Organização do documento

O documento obedece a estrutura a seguir. O Capítulo 2 descreve brevemente o formato do evento e resultados obtidos nos Testes Públicos de Segurança. O Capítulo 3 apresenta em detalhes a seqüência de vulnerabilidades que permitiu aos autores executar um plano de testes que terminou por derrotar o único mecanismo de segurança utilizado pela urna eletrônica para proteger o sigilo do voto. Tambem são descritas alternativas para correção das vulnerabilidades e cenários realistas que ameaçam o caráter secreto do voto, caso as vulnerabilidades não sejam corrigidas. O Capítulo 4 apresenta outro conjunto de fragilidades detectadas no *software* e no processo de desenvolvimento utilizado pelo TSE. Finalmente, o Capítulo 5 apresenta as conclusões e sugestões para que se incrementem a transparência e auditabilidade do sistema eletrônico de votação.

1.4 Agradecimentos

Os autores gostariam de agradecer ao colega Prof. Pedro Rezende da Universidade de Brasília, ao Prof. Dr. Jeroen van de Graaf da Universidade Federal de Minas Gerais, ao Prof. Dr. Paulo S. L. M. Barreto da Universidade de São Paulo e ao Prof. Dr. Francisco Rodríguez-Henríquez do Centro de investigación y de Estudios Avanzados del Instituto Politécnico Nacional de México por discussões relevantes durante a preparação deste documento. Agradecimentos especiais para o Prof. Dr. J. Alex Halderman da Universidade do Michigan por seus comentários extremamente abrangentes em uma versão preliminar deste documento.

Capítulo 2

Testes Públicos de Segurança

Formalmente, o evento teve início com a inscrição das equipes mediante chamada pública de participação e protocolação de documentação correspondente no setor responsável dentro do TSE. Segundo consta no edital de abertura [4], apenas as equipes com inscrições previamente aprovadas pelo Tribunal teriam oportunidade de participar das atividades. A grande novidade da 2ª edição do evento em relação à 1ª foi a possibilidade de se examinar o código-fonte do *software* de votação. A 1ª edição do evento consistiu exclusivamente em testes de "caixa preta".

2.1 Formato

As 9 equipes compostas por 24 investigadores com inscrições aprovadas participaram das duas etapas que compreenderam intervalos de 3 dias, cada um com 10 horas de atividades: (i) fase de preparação, entre os dias 6 e 8 de Março de 2012, quando as equipes puderam examinar o código-fonte do *software* e solicitar esclarecimentos técnicos, em busca de formular hipóteses e planos de teste para avaliação da qualidade dos mecanismos de segurança implementados na urna eletrônica; (ii) fase de testes, entre os dias 20 e 22 de Março de 2012, quando as equipes não mais teriam acesso ao código-fonte e poderiam executar os planos de teste com a finalidade de validar hipóteses e obter conclusões e resultados. Mesmo que os autores não tenham sentido a necessidade de fazer uso desse recurso, é importante registrar que a restrição de acesso ao código-fonte foi relaxada no segundo dia da fase de testes.

As atividades concretas da 2ª edição dos Testes Públicos de Segurança tiveram início no dia 6 de Março de 2012, com a realização de uma palestra de abertura [5] onde foram apresentados o formato do evento e uma visão superficial dos procedimentos para realização de eleições e uma descrição mais detalhada dos mecanismos de segurança implementados na urna eletrônica. O objetivo da palestra foi nivelar o conhecimento dos participantes a respeito do sistema. A equipe de autores, identificada como "Grupo 1", assistiu

atentamente à palestra de abertura para se familiarizar com características técnicas do sistema e começar a detectar pontos de ataque promissores.

No período compreendido entre as duas etapas, foi solicitado que as equipes protocolassem também os planos de teste elaborados a partir das informações coletadas na fase de preparação, visto que apenas as metodologias aprovadas pela Comissão Disciplinadora do evento poderiam ser colocadas em prática posteriormente na fase de testes.

2.2 Objetivos

O edital de abertura ainda discriminou os objetivos das equipes em duas classes bastante distintas de testes, aqui copiadas por completo [4]:

- *Falha*: evento em que se observa que um sistema violou sua especificação por ter entrado em um estado inconsistente e imprevisto ocasionado por uma imperfeição (defeito) em um *software* ou *hardware* impedindo seu bom funcionamento, sem interferir na destinação e/ou anonimato dos votos dos eleitores;
- *Fraude*: ato intencional que tenha alterado informações e/ou causado danos, interferindo na destinação e/ou anonimato dos votos, e que tenha sido efetuado de forma a não deixar vestígios perceptíveis.

Pode-se interpretar a primeira classe como um ataque de *negação de serviço*, onde o atacante tem interesse em impossibilitar a utilização do equipamento pelos eleitores. A segunda classe captura os ataques com potencial de causar fraude eleitoral.

A equipe elaborou e submeteu dois planos de teste, intitulados "Tentativa não-rastreável de quebra do sigilo de votação" [6] e "Tentativa não-rastreável de fraude no resultado de votação" [7], ambos claramente objetivando satisfazer as exigências para uma tentativa de fraude em eleição simulada realizada seguindo procedimentos oficiais. Dado o tempo exíguo, apenas o primeiro plano de testes foi executado por completo.

2.3 Metodologia

A metodologia executada pelo plano de testes exigiu a divisão da equipe em duas partes, aqui identificadas por A e B, que alternavam sua presença no ambiente de testes para impedir qualquer tipo de comunicação. Os experimentos seguiram os procedimentos a seguir:

1. Geração pelo TSE de uma lista secreta de votos fictícios abrangendo os cargos de vereador e prefeito;
2. Entrega da lista secreta de votos para a parte A da equipe;

3. Carga da urna com cartão fornecido pelo TSE e impressão da zerésima;

4. Inserção dos votos na urna eletrônica, seguindo a ordem da lista e efetuada pela parte A da equipe sob monitoração de funcionários do TSE;

5. Rompimento do lacre e entrega da Mídia de Resultados (MR) para a parte B da equipe;

6. Execução de um programa de análise que analisa o Registro Digital do Voto armazenado na MR e produz uma lista de votos em ordem que supostamente representa os votos inseridos na urna;

7. Comparação da lista mantida secreta para a parte B com a lista de votos produzida pelo programa de análise.

O critério de sucesso para a metodologia é naturalmente a correspondência entre as duas listas. Observe que, dentro do ambiente de testes, houve a necessidade de se romper o lacre da MR para se completar a simulação, visto que esta era a única forma de se obter o RDV correspondente à votação simulada. Em eleições reais, o RDV é de acesso público garantido por lei [8]. Naturalmente, também houve a necessidade de se dispor de acesso físico à urna para inserir os votos contidos na lista fornecida pelo TSE, o que foi feito conforme o protocolo descrito acima.

2.4 Resultados

Como consta em relatório final da equipe escrito em conjunto com o TSE [6], a metodologia de avaliação da qualidade dos mecanismos de proteção do sigilo do voto obteve sucesso absoluto em recuperar em ordem os votos de eleições simuladas com 10, 16, 21 e 475 eleitores (20, 32, 42 e 950 votos, respectivamente, já que cada eleitor votava para os cargos de Prefeito e Vereador). Esta última foi realizada para cumprir exigência do TSE em reproduzir a prova de conceito já apresentada para eleições pequenas com uma massa de dados realista, que inclusive reproduz a média de comparecimento das eleições de 2010 no universo de 580 eleitores fictícios utilizado como conjunto de treinamento do evento. Como a metodologia executada consistia apenas em análise dos produtos públicos de votação, não foi necessária nenhuma alteração em qualquer componente da urna eletrônica ou qualquer invasão do perímetro físico do equipamento. Por esse motivo, a metodologia é essencialmente não-rastreável e não deixa nenhum vestígio perceptível.

Registrar os votos inseridos pelo eleitor de forma desordenada é um procedimento crítico para se proteger o sigilo do voto. Fica claro, portanto, que a metodologia da equipe permitiu derrotar o único mecanismo utilizado

pela urna eletrônica para proteger o sigilo do voto. Vale salientar que não foi possível, entretanto, obter uma relação em ordem das identidades dos eleitores a partir unicamente dos produtos públicos de votação, existindo a necessidade de obtenção dessa informação por meios externos para que fosse possível efetuar a correspondência direta e exata entre a identidade do eleitor e sua escolha de candidatos. Até onde se pôde verificar, os produtos públicos de votação armazenam apenas a identificação dos eleitores faltosos em ordem lexicográfica. Discutimos no próximo capítulo como a possibilidade de recuperação dos votos em ordem pode ser utilizada para se montar fraudes eleitorais em cenários realistas.

Não houve tempo suficiente para executar o segundo plano de testes, que objetivava avaliar os mecanismos de segurança que protegem a integridade de uma votação. A prioridade para o primeiro plano de testes foi conferida por sua simplicidade e pela quase completa independência de sua execução de qualquer colaboração significativa por parte do TSE. Efetuar ataques à integridade dos resultados exige que no mínimo se verifique que os resultados fraudados ainda são detectados como autênticos pelos mecanismos de auditoria existentes.

2.5 Pontuação

Foram estabelecidos critérios objetivos de pontuação e uma fórmula para comparação objetiva do desempenho das diversas equipes [9]. Sem justificativas detalhadas e mesmo atingindo absoluto sucesso no teste que se propôs a executar, o plano de testes da equipe recebeu a pontuação ínfima de 0,0313 em uma escala de 0 a 400 pontos [10]. Os critérios para a penalização da pontuação da equipe são absolutamente discutíveis. Não ficou claro, por exemplo, por que foram aplicadas penalidades causadas por pontos de intervenção que só existiam no ambiente simulado dos testes e que não eram obstáculos a uma instanciação do ataque em ambiente real. A Comissão de Avaliação entendeu como quatro os pontos de intervenção: acesso físico à urna, lacre e mídias e visualização do código-fonte. Seria impossível simular qualquer eleição sem nenhum acesso físico à urna e também seria impossível analisar os produtos públicos de votação simulada sem o rompimento do lacre que protege a Mídia de Resultados. O ataque não exigiu acesso ao equipamento além do acesso permitido ao eleitor durante o processo de votação ou ao mesário durante o encerramento da sessão. Vale ressaltar que os partidos políticos recebem o conteúdo da Mídia de Resultados sem a necessidade de qualquer acesso físico a uma determinada urna eletrônica. Além disso, parece incoerente penalizar a equipe por ter visualizado o código-fonte do *software*, quando o objetivo do evento era justamente avaliar a qualidade dos mecanismos de segurança implementados neste. Por fim, a equipe continua sem entender o porquê de sua metodologia ter sido classificada como

uma tentativa de causar falha ao invés de fraude na eleição simulada, visto que em nenhum momento qualquer dos equipamentos utilizados apresentou qualquer defeito. Ainda assim, a equipe consagrou-se como vencedora do evento por obter os resultados mais contundentes e oferecer a contribuição melhor qualificada para aprimoramento da segurança do sistema eletrônico de votação.

Concluímos com duas hipóteses válidas para a pontuação ínfima: ou a Comissão Avaliadora apontada pelo TSE não entendeu a seriedade das conseqüências provocadas pela vulnerabilidade encontrada, ou houve uma tentativa deliberada de descaracterizar e minimizar o ocorrido. Ambas as hipóteses são absolutamente preocupantes.

2.6 Aprimoramento

A participação da equipe nos Testes Públicos de Segurança permitiu ainda coletar algumas sugestões pontuais de aperfeiçoamento do evento em si:

- **Minimização da intervenção do pessoal de apoio**: ainda que seja compreensível a necessidade de se monitorar os investigadores durante a execução dos testes, a falta de privacidade e as intervenções constantes terminaram por causar desconforto à equipe;

- **Redução da burocracia**: novamente, ainda que seja perfeitamente compreensível a necessidade de se registrar todos os procedimentos efetuados pelos investigadores, percebeu-se que satisfazer os requisitos burocráticos exigidos consumiu tempo que poderia ser dedicado à execução de planos de testes adicionais;

- **Ampliação do tempo**: um período de 3 dias é absolutamente insuficiente para se analisar uma porção significativa do código-fonte da urna eletrônica, que em seu total possui milhões de linhas de código. Como em dispotivos de missão crítica *todo* o código termina por se configurar código de *segurança*, visto que uma vulnerabilidade em um trecho inofensivo de código pode disparar um ataque a um trecho crítico de código, é desejável que a duração do evento seja maximizada;

- **Ampliação da capacidade de exame do código-fonte**: foi dedicada uma sala lacrada específica para o exame do código-fonte. Visto que a sala foi equipada com apenas 4 computadores, a falta de capacidade terminou por limitar o tempo de exposição do código-fonte. Em particular, a equipe dos autores só obteve acesso ao código-fonte às 11:00 da manhã do segundo dia da fase de preparação, pois uma das equipes obteve acesso exclusivo à sala no primeiro dia. No total, a equipe dispendiou apenas 5 horas da fase de preparação investigando trechos estratégicos do código-fonte da urna. A disponibilização de

ferramentas simples de processamento de texto (`grep`, `vi`, `cat`, etc) no ambiente de exame de uma base de código de tal tamanho é essencial para a utilidade dos testes e deve ser conservada;

- **Ampliação do escopo dos testes**: o foco do evento foi exclusivamente nos mecanismos de segurança implementandos na urna eletrônica. A justificativa fornecida para tal pode parecer razoável à primeira vista, ao argumentar que qualquer entidade pode fazer uma totalização independente dos resultados, na medida que todos os BUs são publicados na *Internet*. Desta forma, qualquer ataque ao sistema totalizador no máximo atrasaria a apuração e divulgação posterior dos resultados. Entretanto, na opinião da equipe, ataques com sucesso ao totalizador podem forçar ambigüidade ou alteração dos resultados divulgados. Estes podem ser detectados e neutralizados posteriormente apenas em situações onde as respectivas garantias, de que os resultados corretos de cada urna correspondem aos que são publicados na Internet, estejam acessíveis a candidatos potencialmente prejudicados. O sucesso de um ataque dessa natureza pode ainda comprometer a imagem e reputação da autoridade eleitoral ou até mesmo qual o resultado correto das eleições;

- **Aprimoramento dos critérios de pontuação**: a fórmula utilizada para avaliar o desempenho das equipes foi mal-concebida e tinha foco demasiado em aplicar penalidades. Além disso, a apreciação oficial e irrecorrível por parte do Comitê de Avaliação [10] não justificou qualquer de suas decisões, limitando-se apenas a listar os pontos de intervenção e a pontuação final;

- **Alteração da natureza do evento**: o formato atual de competição incentiva as equipes a não compartilhar informações e a utilizar exclusivamente uma métrica de custo/benefício. Ou seja, restrige o escopo a decidir qual das vulnerabilidades encontradas permite modelar um ataque executado na menor quantidade de tempo. Desta forma, várias metodologias sofisticadas perdem prioridade por exigir maior dedicação. Mesmo que essas características modelem uma parcela dos potenciais atacantes, conclui-se que uma avaliação mais criteriosa e colaborativa dos mecanismos de segurança permite a modelagem de atacantes bem-informados e munidos de recursos para executar ataques persistentes.

A avaliação completa e cuidadosa do *software* da urna eletrônica requer enorme esforço e muito tempo de dedicação. Sem a possibilidade de se efetuar testes extensos e sem qualquer tipo de restrição, seguindo metodologia científica, não é possível afirmar que o formato atual do evento colabora significativamente para o incremento da segurança da urna eletrônica. Permite

apenas encontrar vulnerabilidades pontuais que permitam ataques de fácil execução, mas com efeitos limitados.

Capítulo 3

Vulnerabilidades

Neste Capítulo, é descrita a seqüência de vulnerabilidades que permitiu à equipe de autores recuperar os votos em ordem de várias eleições simuladas, uma das quais utilizando um conjunto de dados de tamanho realista.

3.1 Registro Digital do Voto (RDV)

Desde a promulgação da lei eleitoral 9.504/97 [11], que oficializou a votação eletrônica com o modelo atual de urna DRE, o voto impresso verificável pelo eleitor foi instituído no Brasil pela primeira vez em 2002, através da lei 10.408/02 [12]. A finalidade desse recurso é permitir para todos os eleitores, agentes com maior interesse no processo democrático de votação, a possibilidade de verificação independente do seu voto. Sem verificação independente, a confiança é depositada apenas na habilidade dos partidos políticos em fiscalizar a confecção dos programas e na boa fé dos técnicos do TSE em produzir *software* correto [13, página 23]. A proposta do voto impresso sugere produzir uma versão materializada do voto, que pode ser conferida pelo eleitor, sem no entanto permitir que o próprio comprove suas escolhas para uma parte interessada qualquer. Após alegações de dificuldades operacionais e alto custo por parte do TSE, o voto impresso terminou descontinuado pela lei 10.740 em 2003 [8]. Em seu lugar, adotou-se um substituto puramente digital.

O Registro Digital do Voto, ou RDV, é uma tabela separada por cargos em disputa eleitoral que armazena desordenadamente os votos propriamente ditos inseridos pelos eleitores na urna eletrônica. O objetivo desse embaralhamento dos votos é desassociar a ordem em que os votos foram inseridos da ordem em que foram armazenados.

O RDV foi introduzido no lugar do voto impresso para supostamente permitir a mesma capacidade de verificação independente dos resultados da urna. Por essa razão, é um documento público disponibilizado para os partidos após as eleições. Entretanto, enquanto o voto impresso permite

de fato a verificação independente dos votos computados eletronicamente, o RDV é produzido pelo mesmo componente de *software* que produz o Boletim de Urna (BU) contendo os totais de cada candidato computados pela urna. Desta forma, qualquer ataque que comprometa a integridade do BU pode também comprometer o RDV.

Pode-se concluir, portanto, que o RDV não serve a nenhum propósito prático, além de permitir a violação do sigilo do voto caso seja projetado e implementado de forma insegura. A Figura 3.1 apresenta um RDV fictício para uma eleição com 3 cargos com comparecimento de 3 eleitores em um espaço de apenas 7 eleitores possíveis. O primeiro eleitor escolhe 13 para Governador, 31 para Senador e BRANCO para Presidente. O segundo eleitor escolhe 71 para Governador, anula seu voto para Senador com um número inválido e escolhe 37 para Presidente. O terceiro e último eleitor também escolhe 71 para Governador, BRANCO para Senador e 37 para Presidente. Observe que na versão final do arquivo, a princípio não é possível estabelecer correspondência entre a posição do eleitor na ordem de votação e suas escolhas; e que as posições vazias são conservadas pelo embaralhamento.

Governador	Senador	Presidente
	31	
13		
		BRANCO

(a) Registro do primeiro voto.

Governador	Senador	Presidente
71	31	
13		
	NULO	
		BRANCO
		37

(b) Registro do segundo voto.

Governador	Senador	Presidente
71	31	37
	BRANCO	
13		
71	NULO	
		BRANCO
		37

(c) Registro do terceiro voto.

Governador	Senador	Presidente
71	31	37
	BRANCO	
13		
71	NULO	
		BRANCO
		37

(d) Aspecto final do arquivo.

Figura 3.1: Exemplo de armazenamento fora de ordem dos votos no RDV.

3.2 Hipótese

A apresentação desse mecanismo de embaralhamento foi encarada com imediata desconfiança pela equipe após sua apresentação durante a palestra de abertura [5]. A razão para tal foi a constatação de que a ordem em que os votos são armazenados precisa atingir rigor criptográfico de aleatoriedade, e apenas um profissional com algum treinamento básico na área de Segurança Computacional observaria que o mecanismo de embaralhamento é tão crítico para o sigilo do voto quanto a verificação de integridade do *software* para a integridade dos resultados. Desta forma, ainda na palestra de abertura, a equipe levantou a hipótese de que o RDV não havia sido projetado e implementado de forma segura. Com apenas algumas buscas por funções conhecidamente inseguras para a geração de números aleatórios efetuadas na primeira hora de exame do código-fonte, a hipótese já ficou extremamente fortalecida. Restava apenas determinar que informações seriam necessárias para se reverter o embaralhamento e recuperar os votos na ordem em que foram inseridos.

3.3 Projeto e implementação

O mecanismo de embaralhamento foi projetado e implementado utilizando uma progressão de erros que terminou por permitir a sua reversão. A implementação utiliza um *gerador de números pseudo-aleatórios*, procedimento computacional que produz uma seqüência de números aparentemente aleatórios, mas que pode ser unicamente determinada a partir de um pequeno parâmetro chamado *semente* que precisa ser escolhido de forma verdadeiramente aleatória. Quando é necessária a impossibilidade de derivação independente da seqüência por um atacante, a semente deve ser não só apenas verdadeiramente aleatória, mas também mantida em segredo. A seguir, apresentamos a progressão de vulnerabilidades no *software* que forçou o gerador de números pseudo-aleatórios a funcionar fora de seus limites de operação, não atingindo suas propriedades de segurança:

1. **Escolha inadequada de gerador de números pseudo-aleatórios:** foi escolhido o gerador pseudo-aleatório padronizado da linguagem de programação C implementado com as funções `rand()/srand()`. Este gerador possui período curto e aceita sementes muito pequenas, com comprimento de apenas 32 *bits*. Logo, não alcança qualidade criptográfica [14]. A simples ecolha desse gerador pseudo-aleatório já fornece uma metodologia probabilística de ataque;

2. **Escolha inadequada de semente:** a semente consistia em uma tomada de tempo com precisão de segundos no fuso horário UTC (*Coordinated Universal Time*) implementada com a função `time()`

e executada na inicialização do sistema de votação. Esta escolha de semente obviamente não é verdadeiramente aleatória. Como o sistema de votação deve ser inicializado entre as 7 e 8 da manhã do dia de eleição, ainda reduz significativamente o espaço de sementes possíveis em caso de busca exaustiva para apenas 3600 valores;

3. **Publicação da semente:** não obstante a não-aleatoriedade da semente, a mesma ainda era tornada pública mediante registro em LOG e impressão em documento oficial, a zerésima. A zerésima torna-se pública logo após sua emissão, o LOG de eventos torna-se público aos partidos após o final das eleições. De posse da hora de emissão da zerésima, é possível reproduzir a ordem de armazenamento dos votos com exatidão e eficiência, sem probabilidades de erro ou necessidade de busca exaustiva. O mecanismo de assinatura digital do arquivo de LOG e as assinaturas convencionais dos fiscais de partido na zerésima ainda garantem que os documentos são autênticos e que o horário neles contido consiste de fato na semente necessária para se recuperar os votos em ordem.

Os Algoritmos 3.1 e 3.2 apresentam versões simplificadas de como funcionavam a inicialização do gerador pseudo-aleatório e o armazenamento de um voto em posição pseudo-aleatória do RDV, respectivamente. A Figura 3.2 apresenta uma cópia de zerésima encontrada na *Internet*, com a semente (que deveria ser aleatória e secreta) apresentada em destaque. Seja n o números de eleitores que votaram em um espaço total com m eleitores. Da forma como foi projetado e implementado, a presença de posições vazias no arquivo RDV final permite testar valores distintos de semente para obtenção da semente correta sempre que o comparecimento na seção eleitoral não é absoluto. Esse teste é possível pela comparação entre as posições vazias do arquivo final e as posições vazias geradas pela semente sendo testada quando n votos são registrados.

Algoritmo 3.1 Inicialização do RDV.

Entrada: Tabela T representando o RDV, total de m de eleitores.
Saída: Tabela T inicializada e gerador pseudo-aleatório alimentado com semente na forma de tomada de tempo.

1: `srand(time(NULL));`
2: **for** $i \leftarrow 0$ **to** m **do**
3: $\quad T[i] \leftarrow$ VAZIO
4: **end for**

Algoritmo 3.2 Armazenamento de um voto no RDV.

Entrada: Tabela T representando o RDV, número $0 \leq i < n$ do voto V.

Saída: Tabela T atualizada com o voto V inserido.

1: $j \leftarrow$ `rand()` $\mod m$
2: **if** $T[j] \neq$ VAZIO **then**
3: {Colisão encontrada!}
4: Incrementar ou decrementar j até encontrar próxima posição livre
5: **end if**
6: $T[j] \leftarrow V$

Figura 3.2: Zerésima destacando a semente do embaralhamento de votos.

3.4 Ataques

A progressão de vulnerabilidades apresentada na seção anterior permite a formulação de duas metodologias de ataque distintas:

- **Ataque direto:** a partir da semente, que não deveria ser pública mas vinha sendo, recuperada a partir do LOG de eventos ou zerésima da seção eleitoral, é possível simular o embaralhamento de n votos e detectar em que posição cada voto foi armazenado no RDV da seção, que deve ser público e vem sendo, permitindo assim a recuperação dos votos em ordem, a partir de documentos que o atual sistema configura como necessários para a fiscalização do processo eleitoral.

- **Ataque indireto:** a partir dos votos embaralhados, é possível realizar uma busca exaustiva no espaço de sementes possíveis e recuperar a

semente correta pela comparação das posições vazias. De posse da semente correta, o ataque direto pode ser efetuado.

Como os ataques descritos acima não envolvem a modificação ou alteração de qualquer componente do *software* ou *hardware* da urna eletrônica e especificamente não exigem a invasão do perímetro físico de segurança do equipamento, pode-se concluir que ambos são essencialmente não-rastreáveis. A razão para isso é que a leitura de produtos públicos de votação nunca deixa rastro perceptível, visto que não é possível diferenciar a fiscalização da informação pública de um procedimento de ataque ao sigilo do voto. Além disso, os ataques são determinísticos, exatos e reprodutíveis, sem haver qualquer chance de erro. Fica então derrotado o único mecanismo utilizado pelo *software* da urna eletrônica para proteger o sigilo do voto, observando-se o critério de voto secreto como requisito constitucional do sistema de votação. O Algoritmo 3.3 apresenta o ataque direto descrito acima.

Algoritmo 3.3 Recuperação em ordem dos votos do RDV.

Entrada: Tabela T representando o RDV, semente pública s, número n de eleitores que votaram dentre m eleitores possíveis.

Saída: Lista de votos em ordem.

1: `srand`(s)`;`
2: **for** $i \leftarrow 0$ **to** n **do**
3: $\quad j \leftarrow$ `rand()` $\mod m$
4: $\quad$ **if** $T[j] =$ MARCA **then**
5: $\quad\quad$ {Colisão encontrada!}
6: $\quad\quad$ Incrementar ou decrementar j até encontrar $T[j] \neq$ MARCA
7: $\quad$ **end if**
8: $\quad$ Imprimir voto armazenado em $T[j]$
9: $\quad T[j] \leftarrow$ MARCA
10: **end for**

Posteriormente, foi obtida a informação de que o LOG também público de eventos registra o instante de tempo em que cada eleitor confirmou seu voto [15]. Quando esse registro temporal é associado à lista de votos recuperados em ordem, fica também possível recuperar um voto específico inserido em um certo instante de tempo.

3.5 Conseqüências

Agora suponha um atacante capaz de coagir k eleitores e monitorar o comportamento de eleitores no dia de eleição. A recuperação dos votos em ordem permite que esse atacante tenha sucesso com *certeza matemática* em um conjunto de fraudes eleitorais, aqui denominadas por *voto de cabresto digital*:

- Inserção dos eleitores coagidos nas k primeiras posições da fila de votação e posterior recuperação dos k primeiros votos. Isto não parece difícil se o atacante transporta os k eleitores e simplesmente comparece com antecedência à abertura da seção eleitoral;
- Utilização de um voto marcador que indica o início do bloco de k eleitores coagidos. Caso a chegada com antecedência seja um obstáculo, o atacante pode também instruir um eleitor a votar de maneira pré-determinada (anulando seu voto, por exemplo), a partir do qual tem início a seqüência dos votos dos k eleitores coagidos;
- O registro da ordem ou do horário de votação de todos os eleitores da urna eletrônica sob ataque permite a quebra do sigilo do voto para todos os n eleitores que registraram seu voto, mesmo aqueles não coagidos pelo atacante. Observe que essa informação pode ser obtida com colaboração dos mesários ou fiscais de partido.

Um horário de votação específico determina a posição na ordem de votação que um certo eleitor confirmou seu voto. Examinando a posição correspondente na lista de votos recuperada em ordem do RDV revela diretamente quais foram as escolhas de um certo eleitor. Este ataque de *quebra de sigilo direcionado* pode, além de violar o critério de voto secreto assegurado pela constituição [16], causar constrangimento significativo para personalidades públicas (políticos, empresários, industriais, ministros). Note que o local e horário de votação destas personalidades é frequentemente noticiado pela imprensa no dia de eleição [17, 18].

3.6 Correções

A correção da progressão de vulnerabilidades parte do fortalecimento do gerador de posições aleatórias para armazenamento no RDV. Este aprimoramento pode inclusive ser implementado a partir de componentes já disponíveis na própria urna eletrônica. A forma mais segura para se efetuar a correção é substituir o gerador pseudo-aleatório atualmente utilizado por outro gerador pseudo-aleatório com propriedades estatísticas superiores. Exemplos de geradores assim são documentados em padrões [19] e podem ser encontrados em bibliotecas criptográficas de uso geral [20]. Resta apenas determinar fontes de semente para o gerador pseudo-aleatório melhorado. Para satisfazer o critério de aleatoriedade verdadeira, recomenda-se que a semente seja produzida por um gerador em *hardware* baseado em efeito físico bem estudado. Segundo especificação da urna eletrônica modelo 2009 [21], um gerador com estas características já deveria estar disponível no módulo de segurança em *hardware* do equipamento. O processador AMD Geode mencionado na especificação também possui um gerador de números verdadeiramente aleatórios [22] acessível a partir do arquivo `/dev/hw_random`.

Para os modelos anteriores, compromissos de engenharia precisam ser atingidos. Uma solução possível é a obtenção da semente do arquivo `/dev/random`, que comprime eventos de sistema operacional em entropia de qualidade criptográfica acessada por leituras bloqueantes. Problemas para se adotar essa solução envolvem a previsibilidade da inicialização da urna eletrônica, que pode não fornecer a entropia necessária para obtenção de uma semente segura, e o prejuízo de funcionalidade do equipamento por bloqueio da leitura na situação de falta de entropia. A última solução recomendada é relaxar o requisito de qualidade criptográfica e efetuar a obtenção da semente do arquivo `/dev/urandom` a partir de leitura não-bloqueante. Ainda que nesse caso o rigor criptográfico seja perdido, a qualidade do embaralhamento de votos deverá ser significativamente superior à construção atual.

Não custa enfatizar que é preciso realizar testes cuidadosos para determinar se as correções sugeridas atendem aos requisitos mínimos de segurança estabelecidos para o mecanismo de embaralhamento. Os autores se eximem de qualquer responsabilidade caso as soluções sugeridas ainda mantenham o mecanismo de embaralhamento como vulnerável.

Capítulo 4

Fragilidades

O exame do código-fonte do *software* da urna eletrônica não revelou apenas a vulnerabilidade no projeto e implementação do mecanismo de proteção do sigilo do voto, como discutido no capítulo anterior, mas também evidenciou um conjunto de fragilidades em componentes críticos do *software*. Cada fragilidade apresentada aqui representa uma vulnerabilidade em potencial que permite a um agente interno ou externo formular uma metodologia de ataque. A presença de fragilidades, até mesmo em componentes críticos do *software*, atesta a presença de fragilidades no próprio processo de desenvolvimento de *software* utilizado.

4.1 No *software*

A seguir, discutimos as fragilidades encontradas no *software*, algumas já anteriormente discutidas no Relatório elaborado pela Sociedade Brasileira de Computação em 2002 [13], ou na análise acadêmica do *software* de votação das urnas utilizadas nos Estados Unidos e fabricadas pela Diebold [23], mesma companhia que fabrica o *hardware* das urnas brasileiras e produziu as versões iniciais do *software*.

4.1.1 Proteção inadequada ao sigilo do voto

O Registro Digital do Voto (RDV) instituído por dispositivo legal e apresentado no capítulo anterior não fornece nenhuma capacidade de verificação independente adicional, por ser gerado pelo mesmo *software* que realiza a contabilização parcial e gera o Boletim de Urna (BU). Por essa razão, a possibilidade de adulteração do BU implica diretamente na possibilidade de aduteração do RDV, o que significa que o RDV se qualifica apenas como informação redundante, tão passível de ataque quanto aquilo que tenta proteger. Como o RDV não possui qualquer valor prático, serve apenas como uma fonte de ataque ao sigilo do voto caso o mecanismo de embaralhamento

dos votos não seja projetado e implementado de forma segura. Além disso, o próprio projeto da urna não elimina completamente a possibilidade de se vincular a identidade do eleitor ao seu voto através de *software* adulterado [13, página 28], visto que ambos os equipamentos que coletam essas informações estão conectados eletronicamente. É possível concluir que ambas as informações co-existem no estado interno da urna em algum momento e assim podem ser capturadas por um programa malicioso.

Como o RDV foi instituído em 2003, é interessante imaginar se a mesma vulnerabilidade discutida no capítulo anterior estava presente no *software* utilizado nas quatro eleições passadas. Ainda que os autores não tenham atualmente nenhuma intenção de investigar essa possibilidade, existem apenas três respostas possíveis para essa pergunta: (i) o mecanismo de embaralhamento dos votos era *mais* vulnerável; (ii) o mecanismo de embaralhamento era *tão* vulnerável quanto o examinado pela equipe; (iii) o mecanismo de embaralhamento era *menos* vulnerável. As duas primeiras hipóteses indicam que houve proteção inadequada ao sigilo do voto nas quatro últimas eleições, sendo essa propriedade de segurança passível de ataque por agentes internos ou externos com algum conhecimento do mecanismo. A terceira hipótese indica que a qualidade do *software* de votação se degenera com o passar do tempo, existindo a possibilidade do mesmo se tornar menos seguro de uma eleição para outra. As três hipóteses são igualmente preocupantes, especialmente quando se considera que o sigilo do voto é cláusula pétra da Constituição Federal e que o país sempre foi campo fértil para a fraude eleitoral tradicionalmente conhecida como "voto de cabresto".

Recomendação. *Eliminar o RDV e substitui-lo por um mecanismo que forneça a possibilidade real de verificação independente de resultados, como o voto impresso verificável pelo eleitor. Caso a presença do RDV seja uma exigência, recomendamos no mínimo a eliminação das posições vazias do arquivo em seu formato final, para dificultar a busca exaustiva no espaço de sementes possíveis. Caso o mecanismo de embaralhamento dos votos continue frágil, essa compactação não resiste a ataques realizados por agentes internos ou bem-informados.*

4.1.2 Fonte inadequada de entropia

Entropia tem caráter crítico para várias operações criptográficas que requerem dados aleatórios, como a geração de chaves efêmeras ou a alimentação com semente de geradores pseudo-aleatórios, e em muitos casos é possível contornar completamente a técnica criptográfica com ataques apenas na fonte de entropia. Obter entropia suficiente em equipamentos com interatividade limitada a partir unicamente de recursos de *software* é uma tarefa praticamente impossível. Como discutido no Capítulo anterior, o *software* da urna eletrônica brasileira utilizava apenas a medida do tempo em resolução de segundos como fonte de entropia, mesmo tendo disponível fontes

de melhor qualidade em *hardware*.

A coleta de informação previsível para utilização inadequada como entropia não é uma vulnerabilidade desconhecida em sistemas de votação ou *software* comercial. A urna eletrônica utilizava nos Estados Unidos empregava técnicas igualmente inseguras [23, Problema 5.2.12], obtendo informação a partir do conteúdo da tela e de uma medida de tempo com resolução de milissegundo desde a inicialização do sistema operacional. Em 1995, calouros de doutorado da Universidade de Berkeley descobriram sem acesso ao código-fonte que a versão 1.1 do navegador Netscape apresentava exatamente a mesma vulnerabilidade [24], utilizando inclusive a mesma chamada na linha 1 do Algoritmo 3.1.

Recomendação. *Adotar as sugestões apresentadas na Seção 3.6.*

4.1.3 Verificação insuficiente de integridade

A urna eletrônica conta com um mecanismo de verificação de integridade de *software* que tem como objetivo verificar se houve adulteração dos programas da urna eletrônica entre sua a produção e a sua execução propriamente dita no equipamento. Este mecanismo de verificação de *software* muda radicalmente com a presença do módulo customizado de segurança em *hardware*. Por essa razão, a análise será dividida em dois cenários.

Urnas eletrônicas sem módulo de segurança em hardware. A verificação da integridade do *software* cabe a si próprio, podendo ser desativada caso haja acesso livre às porções dos programas que efetuam a verificação. Para mitigar esse risco, costuma-se implementar um mecanismo preliminar de verificação na BIOS (*Basic Input/Output System*), que assegura que o *software* executado em seguida é íntegro. Entretanto, esta técnica apenas reduz a integridade do *software* à integridade do *firmware* programado na BIOS. O problema de verificação de integridade da BIOS, por sua vez, é reduzido a si próprio, sem fonte externa de confiança.

Urnas eletrônicas equipadas com módulo de segurança em hardware. A verificação funciona como descrito anteriormente, com a exceção de que a verificação do conteúdo da BIOS é agora efetuada pelo módulo de segurança. Neste cenário, o problema de verificação de integridade de *software* termina por se reduzir a uma fonte de confiança armazenada no interior do módulo de segurança, sob a forma de uma cadeia autocontida de certificação digital para validação das assinaturas digitais realizadas nos demais componentes de *software*. Derrotar um mecanismo de verificação implementado nesses moldes requer a colaboração de um agente interno capaz de desativar o módulo de segurança ou substituir a cadeia de certificação digital e realizar assinaturas digitais do *software* adulterado utilizando as chaves privadas correspondentes

ao certificado digital inserido posteriormente. Entretanto, segundo a própria especificação do módulo de segurança em *hardware* utilizado nas urnas eletrônicas a partir de 2009, há a necessidade de se programar o módulo de segurança com o resumo criptográfico correto da BIOS [21]. Isto nos leva a concluir que a BIOS transmite o seu próprio resumo para verificação pelo módulo de segurança, ao invés de exigir que o módulo de segurança verifique de forma ativa o conteúdo da BIOS. Assim, uma BIOS maliciosa pode personificar a BIOS legítima, a partir da transmissão do resumo criptográfico correto, e desativar a verificação de assinaturas digitais no caminho seguinte da cadeia de confiança.

Em último lugar, vale ressaltar que os autores observaram que a linha de código no Gerenciador de Aplicativos (GAP), responsável pela verificação de integridade das bibliotecas dinâmicas, encontrava-se desativada com um comentário, atestando que mesmo que a cadeia de confiança seja estabelecida de forma correta, o processo de verificação de integridade ainda é sujeito a sabotagens ou erros de programação.

O Relatório da SBC já apresentava ceticismo explícito a respeito da possibilidade de auto-verificação de *software* através de técnicas criptográficas [13, página 24]. À esta preocupação, soma-se a observação de que garantir que o *software* sendo executado na urna eletrônica é exatamente o mesmo produzido pelo TSE não torna o *software* seguro, apenas confirma sua origem, mesmo quando o mecanismo de verificação de integridade de *software* funciona corretamente.

O problema de verificação de integridade de *software* é endêmico em sistemas de votação eletrônica. Este é um problema particularmente difícil de se resolver na prática. A mesma limitação nos controles de integridade também foi observada no *software* da urna eletrônica utilizada nos Estados Unidos [23, Problemas 4.1.5 e 4.1.6]. É por essa razão que se recomenda a instalação de mecanismos que forneçam capacidade de verificação independente de *software* dos resultados, para que os resultados da eleição não dependam unicamente da integridade do *software*.

Recomendação. *Promover a verificação ativa do conteúdo da BIOS pelo módulo de segurança em* hardware*. Essa recomendação coincide com observações realizadas pelo Grupo 6 durante os Testes Públicos de Segurança [25]. De forma mais geral, recomenda-se transferir a pressão da verificação de integridade do* software *para a verificação independente dos resultados produzidos pelo* software.

4.1.4 Compartilhamento de chaves criptográficas

Todas as urnas eletrônicas em operação no país utilizam a mesma chave criptográfica para cifrar as partições protegidas nos cartões de memória.

O vazamento dessa chave criptográfica tem impacto devastador e revela ao atacante o conteúdo completo dos cartões de memória, incluindo aí o *software* de votação, os mecanismos de verificação de integridade implementados em *software* e a chave privada RSA que realiza a assinatura digital dos produtos públicos de votação [26]. Esta última chave é compartilhada ainda por todas as urnas eletrônicas da mesma unidade federativa [27] e seu vazamento permite a uma atacante produzir um arquivo forjado (LOG, RDV ou BU) mas verificado como autêntico, em nome de uma urna escolhida arbitrariamente. Observa-se que o módulo de segurança em *hardware* introduzido nas urnas eletrônicas possui capacidade ociosa para armazenamento seguro de chaves privadas [21]. Ou seja, o sigilo da chave privada e, conseqüentemente, a integridade dos boletins de urna com a totalização parcial dos votos, reside apenas na confidencialidade de uma chave de cifração compartilhada por meio milhão de equipamentos [5].

Em posição oficial, o TSE argumenta que utilizar chaves múltiplas para cifrar os mesmos arquivos pode revelar alguma característica estatística do texto claro cifrado [28]. Ataques dessa natureza já são estudados na literatura criptográfica, mas não representam nenhum perigo prático relevante [29]. Fica claro, portanto, que este risco nem de perto se compara ao vazamento da chave massivamente compartilhada. A utilização de um modo de operação que aleatoriza o texto claro também elimina trivialmente este risco quando o texto claro não pode ser escolhido arbitrariamente pelo atacante [29], como é o caso discutido aqui.

Recomendação. *Atribuir uma chave criptográfica distinta para cada equipamento, ou pelo menos, para cada cartão de memória utilizado para inseminar um conjunto reduzido de urnas eletrônicas. Mecanismos de derivação de chaves são ferramentas criptográficas projetadas exatamente para resolver este problema.*

4.1.5 Presença de chaves no código-fonte

O compartilhamento da chave de cifração das mídias é agravado pela sua presença às claras no código-fonte do *software*. Isto significa que qualquer agente interno que possua acesso ao repositório onde é mantido o código-fonte também possui automaticamente acesso à chave criptográfica que protege as partições cifradas dos cartões de memória, podendo realizar o vazamento de impacto devastador mencionado anteriormente. Além disso, isto também significa que a chave de cifração faz parte do módulo do sistema operacional responsável por acessar a partição cifrada e tornar disponível seu conteúdo, e por isso precisa estar armazenada às claras dentro do próprio cartão de memória. Ou seja, o objeto cifrado é armazenado no mesmo lugar em que é armazenada a chave criptográfica que o decifra, qualificando este mecanismo como apenas de ofuscação ao invés de verdadeira segurança. Basta que um atacante conheça a posição em que é armazenada a chave de

cifração, por análise da porção do *software* armazenada às claras nos cartões de memória, para que o vazamento da chave se torne possível até para agentes externos.

Recomendação. *Armazenar a chave de cifração no módulo de segurança em* hardware *ou, preferivelmente, em dispositivo criptográfico seguro externo ao ambiente da urna eletrônica.*

4.1.6 Cifração fora dos limites de operação

O algoritmo de cifração das partições protegidas nos cartões de memória é o *Advanced Encryption Standard* (AES) [30] no nível de segurança de 256 *bits*, padrão vigente para cifração em aplicações críticas. O modo de operação selecionado é o *Cipher Block Chaining* (CBC). A combinação de algoritmo e modo de operacão é particularmente recomendada. Entretanto, o modo de operação compartilha, além da mesma chave de cifração, o mesmo *vetor de inicialização*, elemento responsável justamente por aleatorizar o texto claro a ser cifrado e eliminar qualquer propriedade estatística indesejável ao se cifrar o mesmo programa com chaves múltiplas. Escolher de forma uniformemente aleatória um novo vetor de inicialização para cada operação de cifração é requisito inclusive do modo de operação CBC [31]. A argumentação de se utilizar, por alguma questão estatística, uma única chave compartilhada para cifrar todas as mídias de todas as urnas perde completamente o sentido quando o próprio modo de operação de cifração não está sendo utilizado de forma correta e dentro dos limites de operação onde sua segurança é bem entendida.

Recomendação. *Selecionar um novo vetor de inicialização para cada operação de cifração realizada pelo* software *da urna eletrônica, respeitando assim a especificação do modo de operação escolhido.*

4.1.7 Escolha inadequada de algoritmos

Além da escolha absolutamente inadequada de algoritmo para geração de números pseudo-aleatórios, o *software* da urna eletrônica também utiliza a função de resumo criptográfico SHA-1 [32] para fins de assinatura digital e verificação de integridade. Esta função de resumo específica tem uso não-recomendado para essas aplicações desde 2006, quando se verificou que a mesma não fornecia a resistência a colisões esperada, ficando recomendada como prudente a migração *rápida* para funções de resumo mais seguras [33].

Recomendação. *Utilizar um gerador de números pseudo-aleatórios de qualidade criptográfica, como comentado na Seção 3.6, e uma função de resumo criptográfico padronizada e resistente a colisões, como as pertencentes à família SHA-2 [32]. Caso o comprimento da cadeia de caracteres produzida como saída da função de resumo seja crítico para a conferência por humanos, basta utilizar uma função de resumo com segurança superior à*

necessária e truncar o resultado.

4.1.8 Implementações repetidas de primitivas criptográficas

Os autores encontraram diversas implementações repetidas de algoritmos criptográficos ao longo da base de código. A impressão é que cada componente de *software* que utiliza um algoritmo criptográfico recebe sua própria implementação do algoritmo, dificultando em muito a auditoria completa de todas as implementações e aumentando significativamente a chance de erro.

Recomendação. *Concentrar todas as implementações de primitas criptográficas em uma mesma biblioteca crítica de funções que permita fácil auditoria de suas propriedades. Ou ainda, utilizar uma biblioteca criptográfica com reputação estabelecida, como o OpenSSL [20].*

4.2 No processo de desenvolvimento

As fragilidades discutidas na Seção anterior são produto de um processo de desenvolvimento de *software* também frágil. Discutimos a seguir as fragilidades encontradas ou inferidas pelo contexto nesse processo de desenvolvimento. Vários dos mesmos problemas foram também detectados no processo de desenvolvimento da urna utilizada nos Estados Unidos [23, Seção 4.3].

4.2.1 Complexidade acentuada

Segurança advém de simplicidade, transparência e correta avaliação de premissas e condições de confiança. O volume de milhões de linhas de código-fonte empregado para se realizar eleições no Brasil elimina qualquer possibilidade de auditoria completa ou eficaz do *software*. Pode-se argumentar que uma parte significativa dessas milhões de linhas são provenientes do sistema operacional e não precisam de auditoria. Entretanto, verificou-se que o TSE realiza intervenções nos componentes do sistema operacional, para por exemplo inserir a chave criptográfica de cifração no módulo correspondente. Além disso, quando não há compartimentalização suficiente, mesmo vulnerabilidades em trechos inofensivos de código podem criar vulnerabilidades severas em trechos críticos que afetem os mecanismos de segurança.

Um volume de código dessa magnitude irá possuir, *inevitavelmente*, vulnerabilidades que podem ser exploradas. Por essa razão, a base de código deve ser completamente orientada em torno de um pequeno conjunto crítico de funcionalidades, das quais depende o funcionamento correto e seguro do equipamento. Como um valor de referência, os pesquisadores que realizaram a avaliação das máquinas de votar dos Estados Unidos em um intervalo de 60 dias concluíram que os milhares de código dedicados às camadas de aplicação daquele equipamento são de complexidade tal que não é possível tornar o *software* seguro [23, Problema 4.1.2].

Recomendação. *Reduzir o volume de código a partir de técnicas de reuso e componentização, como exemplificado na Seção 4.1.8. Evitar intervenções no código-fonte externo ao TSE e isolar as porções de código de sistema operacional e aplicação para facilitar a auditoria interna do* software.

4.2.2 Auditoria externa insuficiente

Os partidos possuem a prerrogativa legal de examinar o código-fonte do *software* da urna eletrônica, mas para isso precisam assinar um Acordo de Não-Divulgação (AND) que os impede de detalhar publicamente qualquer problema observado no código, mediante imposição legal. Desta forma, os fiscais de partidos são impedidos de prestar contas à sociedade sobre a qualidade do que é feito no *software*, enquanto agentes desonestos possuem toda a liberdade para tentar articular fraudes eleitorais, sem qualquer risco de vazamento dos detalhes das vulnerabilidades encontradas. Como a fiscalização por investigadores independentes é extremamente limitada, consistindo em apenas alguns dias e sob monitoração completa, ou mais recentemente, por um período em que a imensa base de código é constantemente modificada e em condições inadequadas, na prática nenhuma fiscalização efetiva é realizada sobre o *software* do sistema eletrônico de votação. Como afirma de forma contundente o Relatório SBC [13, página 23]:

> *A segurança e corretude dos programas usados na urna baseia-se em confiar na boa fé dos técnicos do TSE. Repetimos: não há nenhuma razão para duvidar da boa fé destas pessoas. Mas isto fere as boas práticas de segurança.*

Como a integridade dos resultados depende unicamente da integridade desse *software*, fica montado um cenário perfeito para fraudes que não deixam vestígios.

Recomendação. *Permitir a auditoria do código-fonte por qualquer cidadão brasileiro, especialista ou não, sem qualquer obstáculo legal.*

4.2.3 Ausência de análise estática de código

A família de funções vulnerável utilizada para embaralhamento dos votos é detectada como potencialmente insegura por qualquer ferramenta de análise estática de código. A ferramenta gratuita *Flawfinder* [34], por exemplo, produz o seguinte alerta quando examina um trecho de código contendo a chamada de função, como por exemplo o programa de análise que implementa o Algoritmo 3.3:

> *This function is not sufficiently random for security-related functions such as key and nonce creation. Use a more secure technique for acquiring random values.*

Recomendação. *Utilizar ferramentas sofisticadas de análise de código para minimizar o impacto de erros de programação que produzem vulnerabilidades, respeitando as boas práticas para desenvolvimento de* software *de missão crítica.*

4.2.4 Formulação equivocada de modelo de atacante

O projeto de mecanismos de segurança utilizado preocupa-se exageradamente com atacantes externos e ignora o risco de atacantes internos. Em particular, como demonstra a própria posição oficial do TSE [28], a detecção de comportamento malicioso por agentes internos é reduzida a processos de auditoria também executados por humanos, obviamente internos. A questão da chave compartilhada de cifração é um exemplo perfeito deste fenômeno, visto que há enorme preocupação com ataques estatísticos esotéricos montados por atacantes externos, enquanto o risco muito maior de vazamento da chave por um agente interno é completamente ignorado. O armazenamento às claras desta mesma chave de cifração dentro da própria urna eletrônica evidencia que os mecanismos de segurança não são projetados para resistir a atacantes que dispõem de informação privilegiada.

Recomendação. *Adotar mecanismos de segurança que resistam a agentes externos e, particularmente, a agentes internos que os conhecem em seus mínimos detalhes.*

4.2.5 Ausência de exercícios internos

Em reunião após a audiência pública para prestação de contas, realizada entre a equipe e vários membros dos setores responsáveis pelas fases de projeto, produção e logística da urna eletrônica, os autores ofereceram a possibilidade de ministrar uma palestra técnica para detalhar todos os problemas encontrados no *software* e o raciocínio específico que os levou à detecção e exploração da vulnerabilidade discutida no Capítulo 3. A proposta foi bem recebida, por permitir aos interessados o entendimento exato de “como funciona a mente do atacante”, nas palavras dos próprios membros do TSE. Não houve convite concreto posterior para tal, mas a leitura dos autores a partir dessa afirmação é de que não existe um time interno responsável por simular o atacante, exercitar metodologias de ataque e tentar derrotar os mecanismos de segurança.

Recomendação. *Instituir, treinar e orientar um time interno de atacantes simulados, prática recomendada para* software *de missão crítica [23]. Não faz sentido projetar mecanismos de segurança sem que existam tentativas simultâneas de subvertê-los.*

4.2.6 Falta de treinamento formal

As fragilidades discutidas nesse Capítulo, presentes inclusive em mecanismos críticos de segurança, demonstram claramente que os membros da equipe de desenvolvimento de *software* do TSE não recebem treinamento suficiente para implementar *software* de segurança. A própria hipótese formulada pelos autores, ainda na palestra de abertura, de que o mecanismo de embaralhamento dos votos poderia não ter sido projetado e implementado de forma segura, por entender que apenas uma equipe com treinamento formal poderia fazê-lo, já demonstra o grau da falta de treinamento observado. A ausência de simulações internas que modelem satisfatoriamente atacantes plausíveis, por falta de entendimento sobre o modo de atuação de um atacante, também corrobora essa observação, visto que um profissional com treinamento adequado na área de segurança já naturalmente costuma se alternar entre os papéis de projetista e atacante por todo o tempo.

Recomendação. *Instituir uma política para treinamento especializado da equipe de desenvolvimento é fundamental para se incrementar a qualidade geral do* software. *Não é plausível esperar* software *seguro como resultado do trabalho de uma equipe de desenvolvimento sem treinamento formal.*

4.2.7 Disponibilização de dados críticos aos investigadores

As máquinas dedicadas por exibir o código-fonte na sala lacrada durante os Testes Públicos de Segurança pareciam ter vindo diretamente da equipe de desenvolvimento. A razão para tal é a disponibilização para todos os investigadores de informações críticas a respeito de nomes de usuário, senhas e o caminho na rede interna para servidores de versionamento do código da urna. Um atacante externo que consiga invadir a rede interna do TSE e esteja munido dessas informações consegue ainda realizar alterações maliciosas no código-fonte e efetivá-las sob a alcunha de um membro legítimo da equipe de desenvolvimento, transferindo completamente para um inocente a responsabilidade por seus atos.

Recomendação. *Sanitizar equipamentos disponibilizados para visitantes externos, para que os mesmos não revelem informações críticas.*

4.2.8 Ignorância da literatura relevante

Como discutido no Capítulo 3, a vulnerabilidade encontrada no embaralhamento dos votos é conhecida há pelo menos 17 anos [24]. Além disso, várias fragilidades apresentadas nesse relatório já foram descritas em laudos técnicos de outros sistemas de votação [23], e mesmo em do próprio [13], os quais contrariam o bom senso e a especificação formal das técnicas criptográficas utilizadas. A persistência desses problemas em uma base de código com 16 anos de história é injustificável e evidencia claramente que a equipe

de desenvolvimento do TSE não acompanha de forma adequada os movimentos relevantes na área de votação eletrônica.

Recomendação. *Responsabilizar parte da equipe de desenvolvimento por estudar e disseminar avanços relevantes de caráter acadêmico ou prático para a segurança de sistemas.*

4.2.9 Falsa sensação de segurança

A repetição incessante de que a urna eletrônica brasileira é absolutamente segura e inviolável, mesmo que isso constitua até uma impossibilidade teórica, perturba o senso crítico dos membros da equipe de desenvolvimento, que terminam por suspender seus próprios mecanismos de avaliação. O processo de desenvolvimento do *software* da urna eletrônica parece funcionar sob o efeito de *suspensão de descrença*, instalando uma falsa sensação de segurança generalizada. Este não é o ambiente ideal para se desenvolver soluções de segurança, especialmente quando as mesmas precisam satisfazer o requisito de missão crítica.

Recomendação. *Adequar o processo de desenvolvimento de* software *para que estimule a verificação mútua e crítica do trabalho realizado, com parâmetros realistas de avaliação.*

Capítulo 5

Conclusões e perspectivas

Este relatório apresentou um conjunto de vulnerabilidades no *software* da urna eletrônica que permitiram a recuperação eficiente, exata e sem deixar vestígios dos votos em ordem registrados eletronicamente. Associando essa informação à lista em ordem de votação dos eleitores, obtida externamente, possibilita a violação completa do sigilo do voto. A partir do registro cronológico de eventos, também é possível recuperar um voto computado em um instante de tempo específico. As conseqüências dessas vulnerabilidades foram discutidas sob um modelo realista de atacante e correções foram sugeridas. Diversas fragilidades adicionais no *software* ou processo de desenvolvimento que o produz também foram detectadas e aqui discutidas, com recomendações concretas de mitigação. Em particular, demonstrou-se como derrotar o único mecanismo de proteção do sigilo do voto utilizado pelo *software* da urna eletrônica.

A necessidade de se instalar recursos para avaliação científica, independente e contínua do *software* torna-se evidente, havendo ampla disponibilidade de especialistas na academia e indústria capazes de contribuir na direção do incremento real das propriedades de segurança na solução adotada para votação eletrônica no país.

Esse conjunto de fragilidades e vulnerabilidades termina apenas por fornecer evidências materiais para as preocupações já levantadas pelo Relatório SBC de 2002 [13, Considerações Finais]. Em particular, pode-se concluir que não houve incremento significativo nas propriedades de segurança fornecidas pelo *software* da urna eletrônica nos últimos 10 anos. Continuam preocupantes a proteção inadequada do sigilo do voto, a impossibilidade prática de auditoria completa ou minimamente eficaz do *software*, e a verificação insuficiente ou inóqua de integridade do *software* de votação. Como estas três propriedades são atualmente críticas para garantir o anonimato e destinação correta dos votos computados, resta aos autores repetir as conclusões do Relatório SBC e defender a reintrodução do voto impresso nos termos apresentados em [13, página 29] como mecanismo simples de verificação de

integridade dos resultados da eleições. O voto impresso distribui a auditoria do *software* entre todos os eleitores, que tornam-se responsáveis por conferir que seus votos foram registrados corretamente pela urna eletrônica, desde que apuração posterior seja realizada para verificar que a contagem dos votos impressos corresponde exatamente à totalização eletrônica parcial. Essa apuração pode ser realizada por amostragem, de forma a não haver impacto significativo na latência para divulgação dos resultados. Vale ressaltar que o voto impresso é para fins de conferência apenas no interior da seção eleitoral e não pode servir de comprovante no ambiente externo à seção eleitoral, como determina a legislação a respeito [35]. A proposta de voto impresso retornaria para o sistema brasileiro de votação nas eleições de 2014, mas infelizmente foi suspensa por alegações questionáveis de inconstitucionalidade.

Um movimento nesta direção acompanharia a tendência mundial vigente em sistemas de votação eletrônica. Com a adoção do voto impresso pela Índia, o Brasil permanece como o único país no mundo a adotar sistema de votação sem verificação independente de resultados. Acreditamos que por esse motivo, e dadas as fragilidades discutidas neste relatório, o *software* utilizado no sistema de votação eletrônica brasileiro não satisfaz requisitos mínimos e plausíveis de segurança e transparência.

Referências Bibliográficas

[1] Janino, G. D.; Balcão Filho, A.; Montes Filho, A.; Lima-Marques, M; Dahab, R.: Relatório do Comitê Multidisciplinar nomeado pela Portaria-TSE 192, 2009.

[2] Stevens, M.: New collision attacks on SHA-1 based on optimal joint local-collision analysis. EUROCRYPT 2013.

[3] Rivest, R. L.; Wack, J. P.: On the notion of "software independence" in voting systems. Philosophical Transactions of The Royal Society A 366 (1881), 2008. Available at http://people.csail.mit.edu/rivest/pubs.html#Riv08b

[4] Tribunal Superior Eleitoral. Edital N° 01/2012. Arquivo disponível em http://www.justicaeleitoral.jus.br/arquivos/tse-2-edicao-dos-testes-de-seguranca-na-urna-eletronica

[5] Azevedo, R.: Aspectos Técnicos da Segurança do Sistema Eletrônico de Votação. Disponível em http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/material/Apresentacao_aspectos-tecnicos.pdf

[6] Grupo 1. Plano de Teste GP1T1 – Tentativa não-rastreável de quebra de sigilo de votação. Disponível em http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/G1PT1.pdf

[7] Grupo 1. Plano de Teste GP1T2 – Tentativa não-rastreável de fraude no resultado de votação. Disponível em http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/G1PT2.pdf

[8] Presidência da República. Lei N° 10.740, de 1° de Outubro de 2003. Disponível em http://www.planalto.gov.br/ccivil_03/leis/2003/l10.740.htm

[9] Tribunal Superior Eleitoral. Edital N° 05/2012. Disponível em http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/TSE-edital-5-2012-criterios-de-classificacao.pdf

[10] Comissão de Avaliação. Avaliações sobre o Teste de Segurança. Arquivo disponível em http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/RelatorioFinal.pdf

[11] Presidência da República. Lei Nº 9.504, de 30 de Setembro de 1997. Disponível em http://www.planalto.gov.br/ccivil_03/leis/l9504.htm

[12] Presidência da República. Lei Nº 10.408, de 10 de Janeiro de 2002. Disponível em http://www.planalto.gov.br/ccivil_03/leis/2002/L10408.htm

[13] van de Graaf, J.; Custódio, R. F.: Tecnologia Eleitoral e a Urna Eletrônica – Relatório SBC 2002. Disponível em http://www.sbc.org.br/index.php?option=com_jdownloads&Itemid=195&task=view.download&catid=77&cid=107

[14] Wheeler, D.: Secure Programming for Linux and Unix HOWTO, 2003. Disponível em http://www.dwheeler.com/secure-programs/Secure-Programs-HOWTO.html

[15] Tribunal Superior Eleitoral. Especificação do Arquivo e Registro de Log das Urnas Eletrônicas para as Eleições 2008, Versão 2. Disponível em http://www.tse.gov.br/internet/eleicoes/arquivos/logs2008/EspecificacaoArquivoRegistroLogUrnasEletronicasEleicoes2008.pdf

[16] Presidência da República. Lei Nº 4.737, de 15 de Julho de 1965. Disponível em http://www.planalto.gov.br/ccivil_03/leis/l4737.htm

[17] Agência de Notícias da Justiça Eleitoral. Presidente do TSE vota em trânsito na capital federal. Disponível em http://agencia.tse.jus.br/sadAdmAgencia/noticiaSearch.do?acao=get&id=1336461

[18] Correio Braziliense. Presidente do TSE, Ricardo Lewandowski, vota em trânsito no IESB. Disponível em http://www.correiobraziliense.com.br/app/noticia/especiais/eleicoes2010/2010/10/03/interna_eleicoes2010,216159/index.shtml

[19] National Institute of Standards and Technology. FIPS 186-1 – Digital Signature Standard (DSS), 1998.

[20] The OpenSSL Project. Disponível em http://www.openssl.org/

[21] Tribunal Superior Eleitoral. Aquisição de Urnas Eletrônicas – UE2009 / PB – Projeto Básico. http://www.tse.jus.br/transparencia/arquivos/tse-projeto-basico-audiencia-publica-2009

[22] AMD. Design without compromise, 2007. Disponível em http://www.amd.com/us/Documents/33358e_lx_900_productb.pdf.

[23] Calandrino, J. A.; Feldman, A. J.; Halderman, J. A.; Wagner, D.; Yu, H.; Zeller, W. P.: Source Code Review of the Diebold Voting System, 2007. Disponível em https://jhalderm.com/pub/papers/diebold-ttbr07.pdf.

[24] Goldberg, I.; Wagner, D.: Randomness and the Netscape Browser. Dr. Dobb's Journal, 1996.

[25] Grupo 6. Plano de Teste GP6T1 – Teste de segurança do sistema eletrônico de votação do TSE. Disponível em http://www.tse.jus.br/hotSites/testes-publicos-de-seguranca/arquivos/G6PT1.pdf

[26] Tribunal Superior Eleitoral. Eleições 2010 – Listagem de Hashs. Disponível em http://www.tse.jus.br/arquivos/tse-urna-eletronica-modelo-2009-eleicoes-2010-turno-1-e-2-atualizado-em-22-09-2010-991ue09

[27] Tribunal Superior Eleitoral. Sistema OKEY - 1º Turno, 2010. Disponível em http://www.tse.jus.br/arquivos/tse-chaves-das-u.f.s-eleicoes-2010-turno-1-e-2-991okey

[28] Coluna Segurança Digital, por Altieres Rohr. Falha na urna brasileira "reproduzia fielmente" erro de 1995, diz professor. http://g1.globo.com/platb/seguranca-digital/2012/05/28/falha-na-urna-brasileira-reproduzia-fielmente-erro-de-1995-diz-professor/

[29] Hong, J.; Sarkar, P.: New Applications of Time Memory Data Tradeoffs. ASIACRYPT 2005: 353-372.

[30] National Institute of Standards and Technology. FIPS 197 – Advanced Encryption Standard (AES), 2001. Disponível em http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[31] National Institute of Standards and Technology. Recommendation for Block Cipher Modes of Operation, 2001. Disponível em http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

[32] National Institute of Standards and Technology. FIPS 180-2 – Secure Hash Standard (SHS), 2002. Disponível em http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[33] National Institute of Standards and Technology. NIST comments on Cryptanalytic Attacks on SHA-1, 2006. http://csrc.nist.gov/groups/ST/hash/statement.html

[34] Wheeler, D.: *Flawfinder*. Disponível em http://www.dwheeler.com/flawfinder/

[35] Presidência da República. Lei N^{o} 12.034, de 29 de Setembro de 2009. Disponível em http://www.planalto.gov.br/ccivil_03/_ato2007-2010/2009/lei/l12034.htm